Multi-Objective Optimization Approaches For Mixed-Model Sequencing On SOC Assembly Line

S. Mahmood Hashemi
Eastern Mediterranean University Famagusta
20-Mersin, Turkey
Northern Cyprus
Hashemi2138@yahoo.com

Afshin Poorkhanali
Iran University of Science and Technology
Narmak, Tehran
Iran
poorkhanali@gmail.com

Abstract: Mixed-model assembly line is a type of production line that it includes a variety of product models similar to product characteristics. There is a set of criteria to judge sequences of products. In this paper, a type of sequencing called as Subject-Of-Class (SOC) defined, then optimum sequencing is presented by using Multi-Objective Optimization. Multi-Objective Simulated Annealing (AMOSA) is used by optimization the criteria of line. Indeed, a release Multi-Objective Imperialist Competitive Algorithm (MOICA) is presented.

Key–Words: Multi-Objective Optimization, SOC, Sequencing assembly line, AMOSA, MOICA

1 Introduction

Design of production system has always been challenging part of industrial engineering. The assembly line is an important part in a manufacturing process and many algorithms are proposed to solve this problem [1, 2, 3, 4]. Assembly lines flow production structure that consists of stations \( k=1, \ldots, m \) arranged along a mechanical material handling equipment and at each station, certain operations are performed. This problem can be visualized with a directional graph which it shows in FIG.1. According to graph of tasks, operations in stations 1...5 must be performed to produce the final result. This graph also shows the precedence of task. It means, tasks that performed at station 1 must be done before stations 2 and 3 start their operations and operations in station 5 is started when operations at both stations 3 and 4 are finished.

The assembly line is the process for allocating the set of tasks to stations. The allocating is done based on the time taken to complete each task [6].

The assembly line can be divided into two problems: 1- balancing between stages and 2- determine the production sequence for different models (products). The first is considered as the assembly line balancing problem. The sequence of introducing models to mixed-model assembly line should be determined considering the goals which are crucial to efficiently implement a just-in-time (JIT) production system [11]. Monden defines two goals for the sequencing problem: 1- leaving the load of the total assembly line for each station on the line and 2- keeping a constant rate of usage for every part used in the line [12]. Handle these goals causes to solve problems of the assembly line. There are some of the papers which try to deal problems of Monden [11].

The assembly line balancing problem (ALBP) is one of the classic problems in industrial engineering and is considered as \( NP\)-hard combinational optimization problem. Optimization can be based on different selected equipment. In previous methods, single-objective optimization was considered, but recent researches are based on multi-objective optimization. In other words, the algorithm tries to allocate the set of a task’s based on various equipment, so it optimizes a number of objectives simultaneously.

SOC production systems partition producing functions into some classes. The system is mixed-model assembly lines and the classes are such as the failure devices, the instability of humans with respect to work rate, skill and motivation. This classification because the operations of producing are divided into metrics which it does not have crisp metrics (such as

Figure 1: Stations Graph
motivation), so we can determine more carefully with this approach. The effective utilization of SOC requires that the following problems be solved:

1. Determination of the sequence classification for producing different products on the line,
2. Determination of the metric distance between classes,
3. Determination of the number and sequence of stations on the line,
4. Determination of line balancing.

Problem 4 can be solved by ALBP significantly, but also goals of this paper consider it. Problems 1 and 3 are determined by a graph. Problem 2 depends on the reader and it can be interpreted by some problems.

The remaining sections of the paper are organized as follows. Section 2 gives a brief explanation of Multi-Objective Optimization problems and presents some algorithms for solving them. The objective of SOC is presented in Section 3. Experimental results of applying multi-objective optimization algorithms on the objectives of SOC in Section 4. Finally, concluding remarks are outlined in Section 5.

2 Multi-Objective Optimization

Most of the problems in the real world are multi-objective in nature and they require multi-objective optimization for better solutions. Multi-objective optimization is defined as [7]:

\[
\begin{align*}
\text{Optimize} & \quad \{ f_1(X), f_2(X), \ldots, f_m(X) \} \\
\text{subject to} & \quad g_i(X) \leq 0, \quad h_j(X) = 0 \quad (1) \\
& \quad i = 1, \ldots, m, \quad j = 1, \ldots, p
\end{align*}
\]

Where \( k \) is number of objective functions, \( X \) is the decision vector, \( m \) is number of inequality constraints and \( p \) is number of equality constraints. Many researchers have tried to find an appropriate approach to solve multi-objective problems [7, 8, 9, 10].

In this paper, two multi-objective optimization algorithms are considered. The first algorithm is AMOSA [14]. AMOSA solves Multi-Objective Optimization problem with just one point and without using any ranking. The second method is MOICA. Imperialist Competitive Algorithm is a known approach for Single-Objective Optimization [15, 16, 17], but MOICA using a population of potential solutions, but by using an archive tries to reach to optimum point.

2.1 AMOSA

A basic concept in Simulated Annealing is an evolution of the solution by simulating the decreasing temperature (\( tmp \)) in a material, where higher the temperature meaning that higher the modification of the solution at a generation. Evolution of the solution is carried at specific temperature profiles. At the first iterations a diverse set of initial solutions for the problem is produced at higher temperatures. And, these solutions are evolved while the temperature decreases to get their local optimums. In multi-objective situation, there are non-dominated solutions, which must be kept in the archive, as a candidate of optimal solutions.

Along the runs of AMOSA algorithm, there are two solutions: \( \text{current-so} \) and \( \text{new-so} \). They can have one of three states compared to each other: i- \( \text{current-so} \) dominates \( \text{new-so} \), ii- \( \text{current-so} \) and \( \text{new-so} \) are non-dominated each other and iii- \( \text{new-so} \) dominates \( \text{current-so} \). If \( \text{new-so} \) is dominated by \( \text{current-so} \), there may be solutions in the archive which dominates \( \text{new-so} \). \( \text{New-so} \) is accepted into the archive by the probability

\[ p = \frac{1}{1 + \exp(\Delta \cdot tmp)} \quad (2) \]

Where \( \Delta \) is differencing between \( \text{new-so} \) and other solutions which dominate \( \text{new-so} \)

\[ \Delta = \frac{\sum_{i=1}^{k} \Delta_i + \Delta}{k + 1} \quad (3) \]

Solutions can escape from local-optima and reach to the neighborhood of the global-optima by this probable acceptance. If \( \text{new-so} \) is dominated by some solutions in the archive, equation (3) is modified to:

\[ \Delta = \frac{\sum_{i=1}^{k} \Delta_i}{k + 1} \quad (4) \]

When \( \text{new-so} \) is non-dominated in an archive by all members, then \( \text{new-so} \) is set as \( \text{current-so} \) and it is added to the archive. If \( \text{new-so} \) dominates some solutions in the archive, then \( \text{new-so} \) is set as \( \text{current-so} \) and it is added to the archive and solutions in the archive which are dominated by \( \text{new-so} \) are removed. If \( \text{new-so} \) is dominated by some solutions in the archive, then equation (2) is changed to:

\[ p = \frac{1}{1 + \exp(-\Delta)} \quad (5) \]

Where \( \Delta \) is the minimum of the difference between \( \text{new-so} \) and dominating solutions is in the
Recent Advances in Information Science

54

is represented by cost value \( k \) function, so if there are \( N \) objective the value of that country according to an tries. The cost of each country is calculated using a 3-dimensional matrix. \( N \) (of values for each country. The population of size \( N \) the number of countries and may be changed during the process; \( N \) probabilistic method. from the archive or from the current population by a it is better to construct the next population randomly global optimum the chance of reaching the algorithm to escape from nondominated solutions. Instead of deterministic methods to discover a large number of serve from diversity.

There are two important points (i) algorithm must keep all non-dominated solutions in an archive to preserve from diversity. (ii) The algorithm must avoid deterministic methods to discover a large number of non-dominated solutions. Instead of deterministic movements, using random movements helps the algorithm to escape from local optimum and increase the chance of reaching the global optimum. Therefore it is better to construct the next population randomly from the archive or from the current population by a probabilistic method.

The first step is generation of the population. Let \( N_{\text{empire}} \) be the initial number of empires, which may be changed during the process; \( N_{\text{country}} \) be the number of countries and \( N_{\text{values}} \) be the number of values for each country. The population of size \( (N_{\text{empire}} \cdot N_{\text{country}} \cdot N_{\text{values}}) \) is randomly generated in a 3-dimensional matrix.

The second step in MOICA is evaluation of countries. The cost of each country is calculated using the value of that country according to an objective function, so if there are \( N_{\text{object}} \) objective functions in the problem and \( k^{th} \) value of country is presented by value\( k \) and cost of \( j^{th} \) country for \( i^{th} \) object function is represented by cost\( j \), total power of \( j^{th} \) country is an array and calculated by:

\[
power_j = \text{cost}_{j1}(\text{value}_1, \ldots, \text{value}_n) + \ldots + \text{cost}_{jn}(\text{value}_1, \ldots, \text{value}_n) \quad (6)
\]

In the third step, the state of each country (colony or imperial) is decided. Countries are partitioned into \( N_{\text{empires}} \) regions. In each empire, countries which are dominated (according to their power) by another one are set as a colony. The method for selecting emperors is randomly and probability of winning of each country in this process is based on dominance count. A non-dominated country is selected as imperial by probability

\[
p(\text{imperial}) = \frac{S_{\text{count}}}{S_{\text{country}}} \quad (7)
\]

Where, \( S_{\text{count}} \) is the number of countries which is dominated by candidate imperial and \( S_{\text{country}} \) is the total number of countries in the empire. The remaining countries in the empire are set as colonies. If there is no non-dominated country in an empire, all countries become a colony. After that, the powers of all colonies are added to the power of emperors, with the probability of imperial. A copy of the non-dominating emperors of each empire is kept in an archive.

In the fourth step, all colonies in all empires move toward the power of their emperors. For this assimilation policy some random values (equal to \( N_{\text{values}} \)) between 0,1 is selected and multiplication of these numbers to difference between power of all countries and power of their emperors adds to their latest power.

In the fifth step, power of countries is re-evaluated, and then emperors are selected once again by the same procedure explained in the previous step. According to MOICA policy, all empires try to take the possession of colonies of other empires and control them. The weakest of all countries is colonized by the most powerful imperial. The number of countries which dominates a particular country determines the weakest colony, which is colonized by the most powerful imperial. The most powerful imperial is decided according to a number of countries it can dominate.

3 Proposed Algorithm

The following variables are defined for the multi-objective SOC sequencing of mixed-model lines.

\( U \): Cumulative deviation of real consumed parts of average consumption.

\( S \): Total number of setup’s classes in a production sequence.
The number of part $j$ is required to assemble one unit of product $i$, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. 

$$a_{ij}:$$ Demand for product $i$. 

$$D_T = \sum_{i=1}^{n} d_k:$$ Total number of units for all products (total demand) also represents number of positions in sequence. 

$$C_q:$$ Class of unique product with $q$ classification (for example cycle time of products or indicator of assembly line), $q = 1, 2, \ldots, Q$. 

$$d(C_q, C_p) = d(C_p, C_q):$$ The distance metric measures between two classes $C_p$ and $C_q$ which denote $d_C(p, q)$, where $p, q = 1, 2, \ldots, Q$. (By definition, $d_C(p, p) = 0$) 

$$\frac{a_{ij}d_i}{D_T}:$$ Average consumption rate of part $j$. 

$$x_{ik}:$$ Total number of units of product $i$ produced over stages 1 to $k$, $i = 1, 2, \ldots, D_T$. 

$$v_{jk} = \sum_{i=1}^{n} a_{ij}x_{ik}:$$ The quantity of part $j$ required during $k$ stages. 

$$s(k):$$ The class number of the position $k$, so that $s(k) = 1, 2, \ldots, Q$, where $k = 1, 2, \ldots, D_T$. 

$$d_0 = \min\{d_C(p, q) > 0 \mid p, q = 1, 2, \ldots, Q\}:$$ The minimum of distance metric measures. If all distance metric measures are zero then $d_0 = 1$. 

The MOP for the mixed-model SOC sequencing problem can be stated as

$$\text{Minimize} \ \{f_1(X), f_2(X)\}$$

where $X$ is the vector of decision variables, i.e. the sequence at which multiple models are introduced to the line, and:

$$f_1(X) = U = \sum_{j=1}^{m} \sum_{k=1}^{D_T} |v_{jk} - k \times r_j|$$

$$St: \sum_{i=1}^{n} x_{ik} = k \ \forall k$$

$$x_{i(k+1)} \geq x_{ik} \ \forall i, k$$

$$x_{ik} \text{ integer} \ \forall i, k$$

$$f_2(X) = S = d_0 + \sum_{k=2}^{D_T} d_C(s(k), s(k-1))$$

Eq.(9) returns cumulative deviation of real consumed parts from its average consumption, which is called $U$. Eq.(10) ensures that in stage $k$, $k$ products are produced, which is equal to total number of units of product from various models. Eq.(11) ensures that total number of units of product $i$ produced over $(k + 1)$ stages are not lower than that of in previous stage. Eq. (12) gives the total number of setup’s classes in a production sequence. If the product in position $k$ is a different class from the product in position $k - 1$, then $d_C$ is required. Otherwise, $d_C = 0$. 

Eq. (8) can be solved by evolutionary optimization algorithms. In this paper, we use two algorithms (AMOSA, MOICA) for this reason. The results of using these algorithms are represented in the following section.

### 4 Experimental Results

Both algorithms (AMOSA, MOICA) are working for 100 generations. In both of them size of the archive is same as size of population. The following results are reached after final generation. In MOICA, the values of the first solution in archive are represented.

<table>
<thead>
<tr>
<th></th>
<th>$F_1$</th>
<th>$F_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMOSA</td>
<td>63</td>
<td>6</td>
</tr>
<tr>
<td>MOICA</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

### 5 Conclusion

In this research, a new definition of JIT problem is presented and named as SOC. The definition of SOC is general, so it can be converted to any special status. For solving the SOC problem, we define two formulas, which must minimize. Two evolutionary algorithms (AMOSA, MOICA) are used for minimization. The results of these algorithms are represented in the Table. 1. Other evolutionary algorithms (such as VEGA, MOGA, NSGA, etc.) can be used for this reason.

References:

[1] Xiaomei Hu, Wenhua Zhu, Tao Yu and Zonghui Xiong, A Script-driven Virtual As-


